Introductions, Binary

Introduction to Computer Systems, Fall 2022

Instructor: Travis McGaha

TAs:

Ali Krema	Andrew Rigas	Anisha Bhatia
Audrey Yang	Craig Lee	Daniel Duan
David LuoZhang	Eddy Yang	Ernest Ng
Heyi Liu	Janavi Chadha	Jason Hom
Katherine Wang	Kyrie Dowling	Mohamed Abaker
Noam Elul	Patricia Agnes	Patrick Kehinde Jr.
Ria Sharma	Sarah Luthra	Sofia Mouchtaris

How are you?

Powered by Poll Everywhere

Start the presentation to see live content. For screen share software, share the entire screen. Get help at **pollev.com/app**

Lecture Outline

- **Introduction & Logistics**
  - **Course Overview**
  - **Assignments & Exams**
  - **Policies**
- Binary
  - Conversions
  - Hexadecimal
  - ASCII
  - Length Constraints

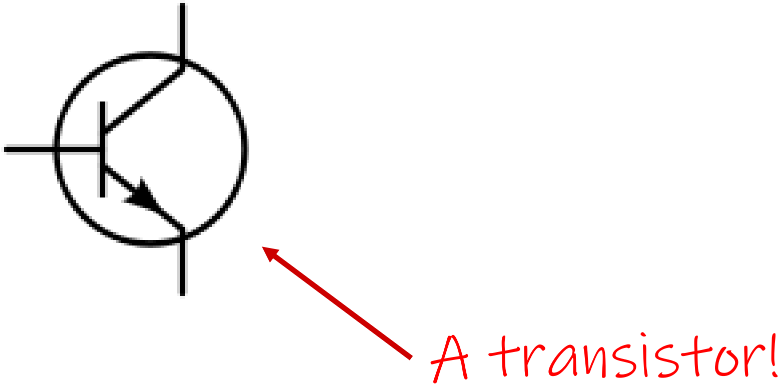
Instructor: Travis McGaha

- UPenn CIS faculty member since… August 2021
  - Currently my third semester at UPenn
  - First Semester Solo with CIS 2400

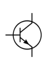
- I have COVID :))))))))))))))))))
  - My brain is not always there

- More on my personal website: https://www.cis.upenn.edu/~tqmcgaha/

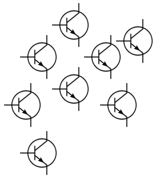
Course Overview: First Half



Course Overview: First Half

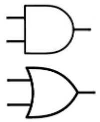


Course Overview: First Half

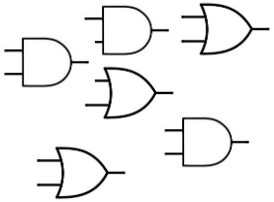


Course Overview: First Half

Logic Gates



Course Overview: First Half



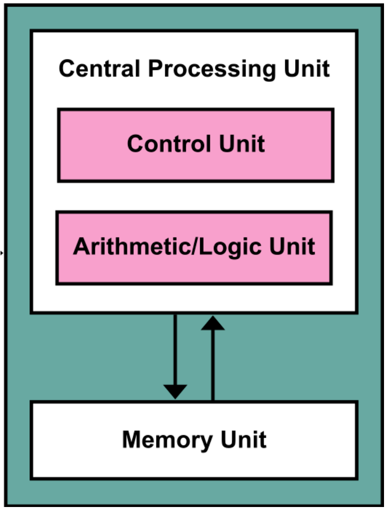
Course Overview: First Half

Adder

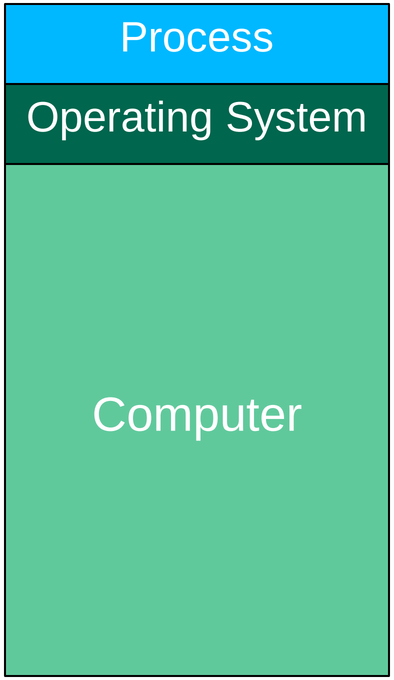
Mux/Demux

Latch/Flip-Flop

Course Overview: First Half



Course Overview: First Half



Course Overview: Second Half

```c
#include <stdio.h>

int main() {
  printf("hello world!\n");
}
```

C Programming ☺

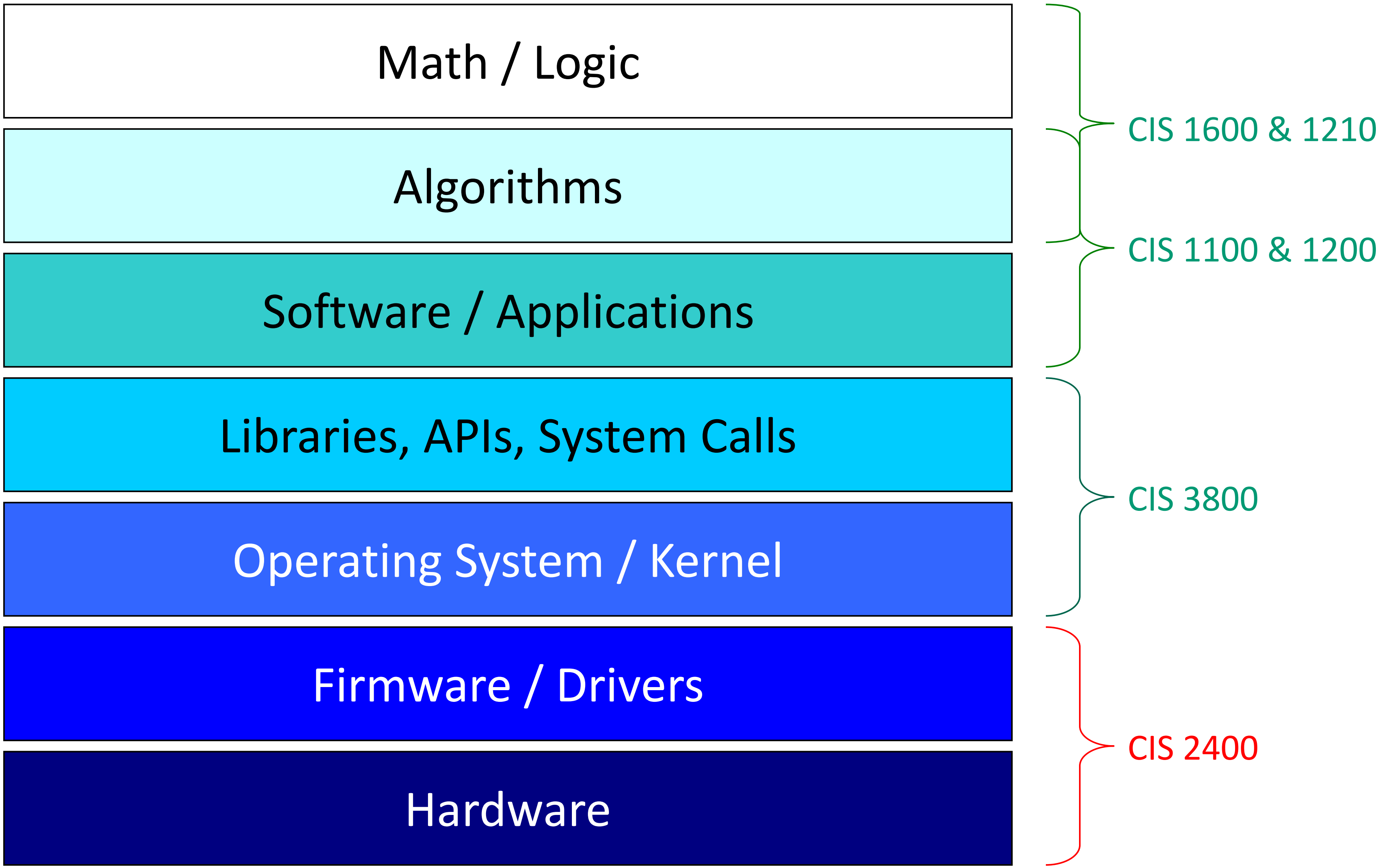
```
.LC0:
        .string "hello world!"
        .text
        .globl  main
        .type   main, @function
main:
.LFB0:
        .cfi_startproc
        pushq   %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset 6, -16
        movq    %rsp, %rbp
```

Assembly Translation

```
00000200: 52e5 7464 0400 0000 b80d 0000 0000 0000
00000210: b80d 2000 0000 0000 b80d 2000 0000 0000
00000220: 4802 0000 0000 0000 4802 0000 0000 0000
00000230: 0100 0000 0000 0000 2f6c 6962 3634 2f6c
00000240: 642d 6c69 6e75 782d 7838 362d 3634 2e73
00000250: 6f2e 3200 0400 0000 1000 0000 0100 0000
00000260: 474e 5500 0000 0000 0300 0000 0200 0000
```

Machine-runnable code

Course Overview



Learning Objectives

- To leave the class with a better understanding of:
  - How a computer “really” works and runs your code
  - What a computer is good at, how to exploit its strengths
  - How modern hardware changes can affect software
  - C programming ☺

- Topics list/schedule can be found on the course website

Prerequisites

- Course Prerequisites:
  - CIS 110/CIS 120

- What you should know already:
  - Vague familiarity with how a program executes
  - Java programming
  - How to write & design large open-ended programs from scratch

Disclaimer

- This is a digest,
- **READ THE WEBSITE**
  - https://www.seas.upenn.edu/~cis2400/current/
- **READ THE SYLLABUS**
  - https://www.seas.upenn.edu/~cis2400/22fa/documents/syllabus

Course Components pt. 1

- Lectures (26)
  - Introduces concepts, slides & recordings available
  - In lecture polling. Polls remain open until the next lecture
- Sections (~10)
  - Reiterates lecture content, lecture clarifications, assignment & exam preparation. Optional, details TBD
- Homework Assignments (12)
  - Due every week
  - Most are programming
  - Very flexible on-request late policy

Course Components pt. 2

- Participation (lots)
  - Lecture polls, Section participation, Weekly Check-in quizzes
- Exams (2)
  - Two in-person exams
  - Midterm will be October 26^{th} “In class”
  - Final will be the week of finals (more details later)
- Textbook (0)
  - No official textbook, but some suggested on course site

Course Infrastructure

- Canvas
  - Grades, surveys, quizzes, Lecture recordings
- Course Website
  - Hosts almost all course content. Syllabus, slides, assignment specifications, course schedule….
  - https://www.seas.upenn.edu/~cis2400/22fa/
- Gradescope
  - Used for most homework turn ins
- Poll Everywhere
  - Used for lecture polls
- Ed
  - Course discussion board

Course Policies

- HW Late Policy
  - Late days given on request
    - (Request usually granted)
  - No cap on the number of late days per assignment
    - More than 3 on an assignment requires approval from Travis
    - Written HWs will not get more than 3 days late.

- Midterm Clobber Policy
  - Final is cumulative
  - If you do better on the “midterm section” of the final, your midterm grade can be replaced.

Getting Help

- Ed
  - Announcements will be made through here
  - Ask and answer questions
  - Sign up if you haven't already!

- Office Hours:
  - Can be found on calendar on top of course website
  - Starts.... soon? (waiting on room reservations)

- 1-on-1's:
  - Can schedule 1-on-1's with Travis
  - Should attend OH and use Ed when possible, but this is an option for when OH and Ed can't meet your needs

We Care

- It is very important that you succeed in CIS 2400 and have a positive experience.
  - Please reach out to course staff if something comes up and you need help
  - Please reach out to course staff if you feel disrespected or uncomfortable by anything

- PLEASE DO NOT CHEAT OR VIOLATE ACADEMIC INTEGRITY
  - We know that things can be tough, but please reach out if you feel tempted. We want to help you succeed
  - Read more on academic integrity in the syllabus

Any questions on anything?

Powered by Poll Everywhere

Start the presentation to see live content. For screen share software, share the entire screen. Get help at **pollev.com/app**

Lecture Outline

- Introduction & Logistics
  - Course Overview
  - Assignments & Exams
  - Policies
- **Binary**
  - **Conversions**
  - **Hexadecimal**
  - **ASCII**
  - **Length Constraints**

Base 10 (Decimal Numbers)

- Humans typically process numbers in base 10
  - Each digit can represent 10 different values
  - Each digit is weighted by its position

- Example: 382

$$(3 * 10^2) \quad + \quad (8 * 10^1) \quad + \quad (2 * 10^0)$$

Base 2 (Binary Numbers)

- Computers typically process numbers in base 2
  - Each "bit" can represent 2 different values (1 or 0)
  - Each "bit" is weighted by its position

- Example: 101

$$(1 * 2^2) \quad + \quad (0 * 2^1) \quad + \quad (1 * 2^0)$$

$$4 \quad + \quad 0 \quad + \quad 1$$

$$5$$

To note that a value is in base 2, a prefix '**`0b`**' is often used

Example: **`0b101`**

Practice

- What is 0b10110 in base 10?

Poll Everywhere

pollev.com/tqm

- What is 0b10110 in base 10?

A. 6

B. 22

C. 16

D. 38

E. I'm not sure

Poll Everywhere

pollev.com/tqm

- What is 0b10110 in base 10?

A. 6

B. 22

C. 16

D. 44

E. I'm not sure

0b10110

$(1 * 2^4) + (0 * 2^3) + (1 * 2^2) + (1 * 2^1) + (0 * 2^0)$

$16 + 0 + 4 + 2 + 0$

$16 + 4 + 2$

22

Decimal to Binary Conversion: Powers of 2

- Algorithm:
  - Subtract the largest power of two <= number
  - Put a one in the corresponding bit position
  - Repeat until number is 0

- Example: 104
  - 104 - 64 = 40 — 64 is 2^6, so bit 6 is a ‘1’
  - 40 - 32 = 8 — 32 is 2^5, so bit 5 is a ‘1’
  - 8 – 8 = 0 — 8 is 2^3, so bit 3 is a ‘1’
  - 104 = 0b1101000

n	2^n
0	1
1	2
2	4
3	8
4	16
5	32
6	64
7	128
8	256
9	512
10	1024

Decimal to Binary Conversion: Division

- Algorithm:
  - Divide by two – remainder will be the next smallest bit
  - Keep dividing until answer is 0

- Example: 104
  - 104 / 2 = 52 r 0 bit 0 is 0
  - 52 / 2 = 26 r 0 bit 1 is 0
  - 26 / 2 = 13 r 0 bit 2 is 0
  - 13 / 2 = 6 r 1 bit 3 is 1
  - 6 / 2 = 3 r 0 bit 4 is 0
  - 3 / 2 = 1 r 1 bit 5 is 1
  - 1 / 2 = 0 r 1 bit 6 is 1
  - 104 = 0b1101000

Poll Everywhere

pollev.com/tqm

- What is 99 in binary?

A. 0b111111

B. 0b110111

C. 0b1011111

D. 0b1100011

E. I'm not sure

n	2^n
0	1
1	2
2	4
3	8
4	16
5	32
6	64
7	128
8	256
9	512
10	1024

Poll Everywhere

pollev.com/tqm

- What is 99 in binary?

A. 0b111111

B. 0b110111

C. 0b1011111

D. 0b1100011

E. I'm not sure

99 – 64 = 35, bit 6 is 1

35 – 32 = 3, bit 5 is 1

3 – 2 = 1, bit 1 is 1

1 – 1 = 0, bit 0 is 1

n	2^n
0	1
1	2
2	4
3	8
4	16
5	32
6	64
7	128
8	256
9	512
10	1024

Hexadecimal

- Base 16 representation of numbers
- Allows us to represent binary with fewer characters
- Prefixes to identify the base
  - 0b11110011 == 0xF3
    ^ **b**inary ^ he**x**
- Hexadecimal will be useful for later homework assignments

Decimal	Binary	Hex
0	0000	0x0
1	0001	0x1
2	0010	0x2
3	0011	0x3
4	0100	0x4
5	0101	0x5
6	0110	0x6
7	0111	0x7
8	1000	0x8
9	1001	0x9
10	1010	0xA
11	1011	0xB
12	1100	0xC
13	1101	0xD
14	1110	0xE
15	1111	0xF

Poll Everywhere

pollev.com/tqm

- What is 0b110101110100 in hex?

A. 0xD74

B. 0x6BA

C. 0x45D

D. 0x2EB

E. I'm not sure

Decimal	Binary	Hex
0	0000	0x0
1	0001	0x1
2	0010	0x2
3	0011	0x3
4	0100	0x4
5	0101	0x5
6	0110	0x6
7	0111	0x7
8	1000	0x8
9	1001	0x9
10	1010	0xA
11	1011	0xB
12	1100	0xC
13	1101	0xD
14	1110	0xE
15	1111	0xF

Poll Everywhere

pollev.com/tqm

- What is 0b110101110100 in hex?

A. 0xD74

B. 0x6BA

C. 0x45D

D. 0x2EB

E. I'm not sure

0b110101110100

0xD 0x7 0x4

Decimal	Binary	Hex
0	0000	0x0
1	0001	0x1
2	0010	0x2
3	0011	0x3
4	0100	0x4
5	0101	0x5
6	0110	0x6
7	0111	0x7
8	1000	0x8
9	1001	0x9
10	1010	0xA
11	1011	0xB
12	1100	0xC
13	1101	0xD
14	1110	0xE
15	1111	0xF

Hex Spelling

- **`0xDEADC0DE`**
- **`0xDEADBEEF`**
- **`0xBO1DFACE`**
- **`0xBADA55`**
- **`0xCAFEF00D`**
- **`0xF00`**

Encoding

- We can represent more than just numbers with bits
  - We just need an agreed upon encoding

- Decimal Numbers
  - 0 → **0x00**, 1 → **0x01**, … , 240 → **0xF0** …

- Characters
  - A → **0x41**, B → **0x42**, C → **0x43**, …

- Colors
  - → 0x281EF2, → 0x990000

The Meaning of Bits

- *A sequence of bits can have many meanings!*

- Consider the hex sequence 0x4E6F21
  - Common interpretations include:
  - The decimal number 5140257
  - The characters “No!”
  - The background color of this slide
  - The real number 7.203034×10^{-39}

- A series of bits can also be code!

- It is up to the program/programmer to decide how to ***interpret*** the sequence of bits

ASCII

- We can encode binary values to represent characters

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

ASCII Design

- ASCII:
  American Standard Code for Information Interchange

- Designed to communicate American letters, numbers, and some control signals efficiently
  - Used only 7 bits to minimize number of bits that need to be communicated
  - Other languages not considered

Unicode

- Unicode Standard UTF-8 is an alternate text encoding
  - Uses between 8 and 32 bits for each “character”
  - Characters include more than just English
  - Characters include emojis

- Unicode table is a lot longer: https://unicode-table.com/en/

Aside: Length Terminology

- Bit:
  - a binary "digit", either a 1 or a 0
- Byte:
  - 8 bits
  - E.g., 0b11110111 or 0xF7
- Nibble:
  - 4 bits
  - E.g., 0b1011 or 0xB

Data Lengths

- Computers are physical machines
  - there is a limit to how many bytes we can store

- In C:
  - `int`'s are usually 4 bytes
    - 4 bytes = 32 bits → 2^{32} different values
    - 2^{32} = 4,294,967,296
  - `char`'s are usually 1 byte
    - 1 byte = 8 bits → 2^{8} different values
    - 2^{8} = 256

Lecture Take-aways

- Bits are the “atom” of data for computers

- We can represent anything in binary by using different encodings!
  - Numbers, colors, characters, emojis, code, etc..

- Our encodings/data is limited due to finite bits
  - (More on this next time)